

We claim:

1. A method for growing a hot trace in a program during the program's execution in a dynamic translator, comprising the steps of:
 - identifying an initial block; and
 - starting with the initial block, growing the trace block-by-block by applying static branch prediction rules until an end-of-trace condition is reached.
2. A method for growing a hot trace in a program during the program's execution in a dynamic translator, comprising the steps of:
 - identifying an initial block as the first block in a trace to be selected;
 - until an end-of-trace condition is reached, applying static branch prediction rules to the terminating branch of a last block in the trace to identify a next block to be added to the selected trace; and
 - adding the identified next block to the selected trace.
3. The method as defined in claim 2, in further comprising the step of storing the selected traces in a code cache.
4. The method of claim 2, in which the end-of-trace condition includes at least one of the following conditions:
 - (1) no prediction rule applies; (2) a total number of instructions in the trace exceeds a predetermined limit; (3) cumulative estimated prediction accuracy has dropped below a predetermined threshold.
5. The method as defined in claim 2, in which the prediction rules include both rules for predicting the outcomes of branch conditions and for predicting the targets of branches.
6. The method as defined in claim 2, in which an initial block is identified by maintaining execution counts for targets of branches and when an execution count

exceeds a threshold, identifying as an initial block, the block that begins at the target of that branch and extends to the next branch.

7. The method claim 2, wherein said set of prediction rules comprises:

for the branch instruction, determining whether to add a target instruction of the branch instruction to the hot trace based on said set of static branch prediction rules.

8. The method as defined in claim 7, wherein said set of static branch prediction rules comprises:

determining if said branch instruction is unconditional; and

if said branch instruction is unconditional, then adding the target instruction of the branch instruction and following instructions through the next branch instruction to the hot trace.

9. The method as defined in claim 7, wherein said set of static rules comprises:

determining if a target instruction of said branch instruction can be determined by symbolically evaluating a branch condition of said branch instruction; and

if said target instruction of said branch instruction can be determined symbolically, then adding the target instruction and following instructions through the next branch instruction to the hot trace.

10. The method as defined in claim 7, wherein said set of static rules comprises:

determining if a heuristic rule can be applied to said branch instruction; and

if a heuristic rule can be applied to said branch instruction, then the branch instruction is determined to be Not Taken.

11. The method as defined in claim 9, wherein said set of static branch prediction rules comprises:

determining if a heuristic rule can be applied to said branch instruction; and

if a heuristic rule can be applied to said branch instruction, then the branch instruction is determined to be Not Taken.

12. The method as defined in claim 10, further comprising the step of changing a count in a confidence counter if said heuristic rule can be applied to the branch instruction; and determining whether said confidence counter has reached a threshold level.

13. The method as defined in claim 7, wherein said set of static rules comprises:
determining whether said branch instruction is a procedure return; and
if said branch instruction is a procedure return, then determining if there has been a corresponding branch and link instruction on said hot trace;
if there has been a corresponding branch and link instruction, then determining if there is an instruction in the hot trace between said corresponding branch and link instruction and the procedure return that modifies a value in a link register associated with the corresponding branch and link instruction; and
if there is no instruction that modifies the value in said link register between said corresponding branch and link instruction and the procedure return, then adding an address of a link point and following instructions up through a next branch instruction to the hot trace.

14. The method as defined in claim 11, wherein said set of static rules comprises:
determining whether said branch instruction is a procedure return; and
if said branch instruction is a procedure return, then determining if there has been a corresponding branch and link instruction on said hot trace; and
if there has been a corresponding branch and link instruction, then determining if there is an instruction in the hot trace between said corresponding branch and link instruction and the procedure return that modifies a value in a link register associated with the corresponding branch and link instruction; and
if there is no instruction that modifies the value in said link register between said corresponding branch and link instruction and the procedure return, then adding an address of a link point and following instructions up through the next branch instruction to the hot trace.

15. The method of claim 13, further comprising the steps:

storing a return address in a program stack;

wherein said step of determining if there is an instruction that modifies the value in the link register comprises forward monitoring hot trace instructions between the corresponding branch and link instruction and the return for instructions that change a value in a link register associated with said corresponding branch and link instruction.

16. The method of claim 2, further comprising a confidence count that is incremented or decremented by a predetermined amount based on which static branch prediction rule has been applied; and

if said confidence count has reached a second threshold level, ending the growing of the hot trace.

17. The method of claim 2, wherein said identifying an initial block step comprises associating a different count with each different target instruction in a selected set of target instructions and incrementing or decrementing that count each time its associated target instruction is executed; and

identifying said target instruction as the beginning of said initial block if the count associated therewith exceeds a hot threshold.

18. The method of claim 17, wherein said selected set of target instructions includes target instructions of backwards taken branches and target instructions from an exit branch from a trace in a code cache.

19. The method of claim 2, wherein the end-of-trace condition comprises when a total number of instructions in the trace exceeds a predetermined limit.

20. A dynamic translator for growing a hot trace in a program during the program's execution in a dynamic translator, comprising:

first logic for identifying an initial block as the first block in a trace to be selected;

second logic for, until an end-of-trace condition is reached, applying static branch prediction rules to the terminating branch of the last block in the trace to identify a next block to be added to the selected trace; and

third logic for adding the identified next block to the selected trace.

21. A computer program product, comprising:

a computer usable medium having computer readable program code embodied therein for growing a hot trace in a program during the program's execution in a dynamic translator, comprising

first code for identifying an initial block as the first block in a trace to be selected;

second code for, until an end-of-trace condition is reached, applying static branch prediction rules to the terminating branch of the last block in the trace to identify a next block to be added to the selected trace; and

third code for adding the identified next block to the selected trace.